Text Analysis Tools and Techniques of the PubMed Data Using the Titan Scalable Informatics Toolkit

Andrew T. Wilson, Michael W. Trahan, Jason F. Shepherd, Thomas J. Otahal, Steven N. Kempka, Mark C. Foehse, Nathan D. Fabian, Warren L. Davis IV, Gloria Casale

Abstract—The PubMed dataset is a collection of documents archiving the emergence of growing biomedical research communities spanning over 127 years. This document collection contains over 17 million citations rendering common data retrieval and analytic techniques infeasible. In this paper, we outline several experimental results demonstrating high performance computing resources and analytic algorithms aimed at enabling search and generalized queries over the entire corpus. These methods include interfaces for parallelized and interactive queries, large scale text analytic runs using latent semantic and latent Dirichlet methods on the entire corpus, topological clusterings of relationship graphs, and high-end visualizations of results using data spanning the entire corpus. We discuss potential avenues for future efforts and feasibility of these efforts in yielding relevant information and relationships for guiding more focused queries of the corpus.

Index Terms—Visualization, PubMed, Text Analysis, Informatics, Analytics

1 Introduction

The PubMed repository is a collection of documents spanning over 127 years of medical research. This collection occupies a unique niche in research history. Within its body, the stories of the emergence of growing research communities can be found – with the right set of tools. In this report we document initial informatics research efforts to probe this document collection utilizing the Titan Scalable Informatics Toolkit. We couple the Titan algorithms with the VisTrails system to provide improved user interaction, flexibility and provenance tracking capabilities within an end-user application. In this paper e present results using high performance computing algorithms for data retrieval and analytics and provide some comparison and initial benchmarking of these capabilities to bound our expectations as the research continues forward.

Contributions This paper makes the following contributions to the research community:

- We present an outline of capabilities housed within the Titan toolkit for informatic analysis of large document caches.
- We present a coupled environment for user interaction using the Titan toolkit and the VisTrails workflow and provenance management system.
- We demonstrate tests and scalability studies for latent semantic analysis (LSA) and latent Dirichlet allocation (LDA) pipelines using the ParaText application in Titan.
- We describe VisTrails packages for broad-based queries against the PubMed data.
- We show several visualizations highlighting the scalability and flexibility of the Titan tools and algorithms for accessing and addressing large data sets.

This paper is organized as follows: we provide some brief background and statistics on the PubMed document collection, as well as capability descriptions for the Titan toolkit and the VisTrails application in Section 2. In Section 3, we discuss the ParaText application within Titan and the high performance computing environment utilized to provide the testing and scalability studies. Section 4 provides discussion on the specific data analytics tasks performed. In section 5, we examine results of the PubMed analyses and make recommendations for improvements. Sections 6 and 7 present some future directions for this work and concluding remarks, respectively.

2 Background and Related Works

Informatics and analytics are rapidly growing areas of research in the scientific, academic and commercial realms. With available data growing at exponential rates, tools to effectively analyze the wealth of data are in high demand. In this section, we discuss the PubMed dataset

- *Andrew T. Wilson is with Sandia National Laboratories, E-mail: atwilso@sandia.gov*
- *Michael W. Trahan is with Sandia National Laboratories, E-mail: mwtraha@sandia.gov*
- *Jason F. Shepherd, Member IEEE, is with Sandia National Laboratories, E-mail: jfsheph@sandia.gov*
- *Thomas J. Otahal is with Sandia National Laboratories, E-mail: tjotaha@sandia.gov*
- *Steven N. Kempka is with Sandia National Laboratories, E-mail: snkempk@sandia.gov*
- *Mark C. Foehse is with Sandia National Laboratories, E-mail: mcfoehs@sandia.gov*
- *Nathan D. Fabian is with Sandia National Laboratories, E-mail: ndfabia@sandia.gov*
- *Warren L. Davis IV is with Sandia National Laboratories, E-mail: wldavis@sandia.gov*
- *Gloria Casale is with Sandia National Laboratories, E-mail: gcasale@sandia.gov*

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energys National Nuclear Security Administration under contract DE-AC04-94AL85000.

and provide some high level statistics and descriptions of the data contained in this collection. We will also provide capability descriptions of the Titan toolkit and the VisTrails applications which were utilized in this effort.

2.1 The PubMed document collection

PubMed is a free database of citations, abstracts, and some full text articles of life science and biomedical topics provided by the U.S. National Library of Medicine. The PubMed document collection is a historical trove of primarily biomedical research spanning the past 127 years. The dataset contains references for 17 million articles, of which 10.5 million have included abstracts. The dataset references 9.3 million authors, 831 thousand funding sources, 1.5 million journal issues, and 6.3 million institutions. While each of the articles contained in the collection is unique, the technical nature of the documents highlight textual difficulties that are also unique. For instance, it is difficult to eliminate overlap due to mis-spellings, alternate naming conventions, empty fields, hyphenation and acronym synchronization to get a completely accurate representation of the other corpus-specific statistics.

2.2 The Titan Toolkit

The Titan Informatics Toolkit is a collaborative effort led by the Scalable Analysis and Visualization Department at Sandia National Laboratories. The Titan project is a significant expansion of the Visualization ToolKit (VTK) [18] to support the ingestion, processing, and display of informatics data. VTK is the core engine for many of the scientific visualization tools developed at the U.S. Department of Energy's National Laboratories, including both the ParaView [17] and VisIt [19] projects. Historically, the Titan project represents one of the first software development efforts to systematically address the merging of scientific visualization and information visualization on a substantive level.

VTK utilizes a dataflow paradigm for development of modular and reusable code components. The dataflow model enables flexible configuration of algorithms into pipelines. By leveraging this structure, Titan provides a flexible, component-based pipeline architecture for the integration and deployment of algorithms in the fields of cyber data, semantic graph and information analysis (see Figure 1). VTK also provides a parallel client-server layer, making it an excellent framework for performing scalable data analysis on distributed memory platforms. The combination of algorithms and architectures for visualization and analysis of both scientific simulations and informatics data is already proving its worth in the form of functionality, flexibility and breadth across multiple disciplines.

Titan also integrates the capabilities of a series of highly functional open-source toolkits, including graph algorithms (Boost Graph Library [23]), linear algebra (Trilinos[22]), named entity recognition (StanfordNER[24, 13]), MatLab [20, 14], 'R' statistics [16], ProtoVis [4], clustering (GMeans [11, 8, 10, 9]) and others. Titan components are written in C++, as are new additions to the toolkit. These components can be used directly from C++ or through automatically-generated bindings for Python, Java and Tcl.

Titan includes a growing list of algorithms and helper classes. At the time of this writing, the set of tools available include:

- Readers - Dimacs, DOT, GXL, Chaco, XML, Tulip, CSV, fixed-width and delimited text (including Unicode), ISI, RIS, Palantir XML, OWL, JSONPlus, PDF (using xpdf), and MS Word (using antiword)

- Database Connectors - MySQL, PostgreSQL, Oracle, SQLite, ODBC and others

- Graph Algorithms - biconnected components, Brandes centrality, breadth-first search, connected components, minimum spanning tree, wCNM community detection, CSG search, and S-T search

- Linear Algebra - MapReduce, TPP/PARAFAC, QR decomposition, SVD, CCA and kCCA

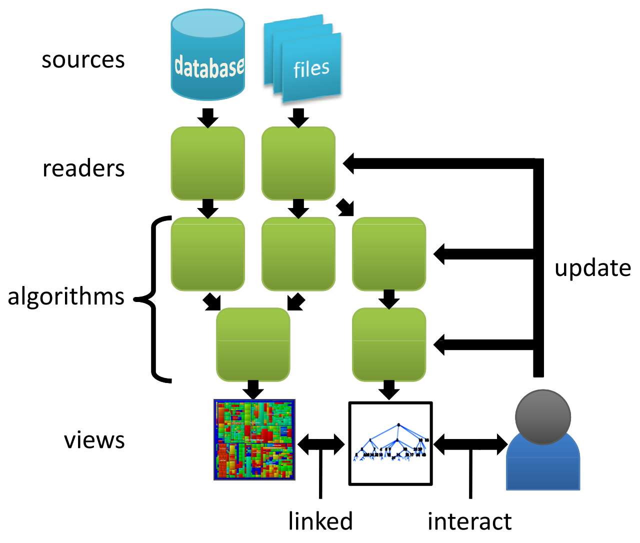


Fig. 1. The Titan workflow paradigm. Data is ingested, modified, processed and eventually visualized utilizing a pipeline framework.

- Graph Layout Algorithms - GSpace, user-assigned coordinates, circular layout, clustered, community-clustered, cone, constrained, cosmic tree, fast 2D, force-directed, random, Fruchterman and Reingold (simple2D), space-filling curve, tree layout (standard and radial), tree orbit, tree ring, tree map, and icicle layout

- Text Analysis Algorithms - LSA, LDA, entity extraction, MIME type detection, case folding, frequency matrix weighting and filtering, n-gram extraction, XML processing, tokenization, dictionaries, cosine and Jensen-Shannon similarity computation

- Web Integration - Apache module support, JavaScript, Protovis, and JQuery

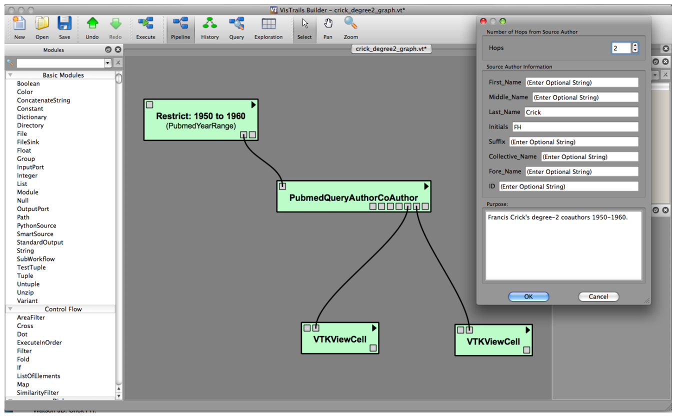
- Statistics Algorithms - descriptive, order, correlative, contingency and multi-correlative statistics, PCA, and K-Means clustering

- Views - Graph, hierarchical graph, tree, treemap, geo-referenced, parallel coordinates, tree ring, icicle, images, etc.

2.3 VisTrails

VisTrails[25] is an open-source provenance management and scientific workflow system that was designed to support the scientific discovery process. VisTrails provides unique support for data analysis and visualization, a comprehensive provenance infrastructure, and a user-centered design. The system combines and substantially extends useful features of visualization and scientific workflow systems. The availability of provenance information enables a series of operations that simplify exploratory processes and foster reflective reasoning. For example, scientists can easily navigate through the space of workflows created for a given exploration task, visually compare workflows and their results, and explore large parameter spaces. An example of the VisTrails user interface is shown in Figure 2.

VisTrails provides a plugin infrastructure to integrate user-defined functions and libraries. Specifically, users can incorporate their own visualization and simulation codes into pipelines by defining custom modules (or wrappers). These modules are bundled into *packages*. A VisTrails package is simply a collection of Python classes where each of these classes represents a new module.



Number of Hops from Source Author

Hops: 2

Source Author Information

First_Name: (Enter Optional String)
Middle_Name: (Enter Optional String)
Last_Name: Crick
Initials: FH
Suffix: (Enter Optional String)
Collective_Name: (Enter Optional String)
Fore_Name: (Enter Optional String)
ID: (Enter Optional String)

Purpose:
Francis Crick's degree-2 coauthors 1950–1960.

Fig. 2. The VisTrails interface showing a pipeline, or workflow, view. Data 'flows' between modules. Each module contains an algorithm that modifies the data, and individual algorithmic parameters can be accessed through the module interface (a sample interface is also shown). The VisTrails interface provides a more intuitive option for interacting with algorithms available in packages like the Titan Toolkit.

VisTrails was designed to manage rapidly-evolving workflows. The provenance infrastructure maintains detailed history information about the steps followed and data derived in the course of an exploratory task. The system also provides extensive annotation capabilities that allow users to enrich the automatically captured provenance. This information persists as XML files or in a relational database. Besides enabling reproducible results, VisTrails also aids collaborative analysis by presenting an entire workflow as an artifact to be modified and explored through an intuitive user interface. The system supports reflective reasoning by storing temporary results, by providing users the ability to reason about these results and to follow chains of reasoning backward and forward. Users can navigate workflow versions in an intuitive way, undo changes without losing results, visually compare multiple workflows and display their results side-by-side in visual spreadsheets, and examine the actions that led to the result.

3 High Performance Computing

As data sets grow toward petabytes, high performance computing is becoming a favored tool for analytics across business, academic and government institutions. Indeed, as data sets grow ever larger, high-performance high-capacity systems will become a *required* tool. In this section we describe the parallel platforms, tools and algorithms that form key components of our work.

3.1 Red Sky

Red Sky is a high performance computing capacity machine at Sandia National Laboratories consisting of 2,318 nodes, each outfitted with two quad-core CPUs (18,544 cores total) delivering 217 Teraflops of computing. The CPUs are 2.93 GHz quad-core Nehalem X5570 processors with a 3D torus Infiniband interconnect. Each node has 12GB of shared RAM (1.5GB per core). The operating system is CentOS, a RedHat-based Linux kernel with patches.

3.2 Database System

For this project, the PubMed data set was stored in a relational database on a Netezza(r) Performance Server 10050. The NPS [1] is a parallel database system that partitions data sets and SQL queries across a collection of storage and computation nodes. Queries are similarly partitioned across the system with any necessary data interchange handled transparently. An ANSI SQL interface with optional extensions for user-defined analysis functions is provided for interaction. This standard, commodity-protocol interface is very valuable for this project both because of its familiarity (which speeds integration) and its wide adoption (which allows substitution of other implementations). For instance, the entire system can be alternatively deployed on a single, stand-alone workstation using almost any commercial or open-source SQL database, e. g., if we need to isolate a sensitive data set.

On larger data sets such as PubMed, the capabilities of a parallel database appliance become a core enabling capability. Although it is certainly possible to design and implement an optimal out-of-core algorithm for any given task, the wide variety of queries we need to execute renders this approach inefficient. This is doubly true in an analysis setting where answers must often be available promptly to be of any value at all.

3.3 ParaText

ParaText is a system for analyzing document caches through a 'bag-of-words' type of text analysis. ParaText is specifically designed for scalable distributed memory analysis of large document collections. The ParaText pipeline includes several text analysis components coordinated within a Titan data processing pipeline where data sources, filters, and sinks can be combined in arbitrary ways . ParaText performs a full Latent Semantic Analysis (LSA) or Latent Dirichlet Allocation (LDA) process including document ingestion from a number of sources and formates, text extraction and tokenization, term-document frequency matrix formulation and weighting, leading finally to the LSA or LDA analysis itself.

ParaText is built as a dataflow model with all of these algorithms linked together into data-parallel pipelines that can be replicated on each core of a distributed-memory architecture. Individual components within the pipeline can be replaced or rewired to explore different computational strategies or demonstrate new functionality.

3.3.1 Document Ingestion

ParaText can use any of the Titan data readers to load documents from a wide variety of file formats. It can also connect to a database to ingest data as the result of an SQL query. Once documents are in memory they undergo text extraction, tokenization (including n-gram extraction), dictionary construction, construction of a term/document frequency matrix and then filtering to remove stop words and unwanted tokens. This entire process is performed in parallel.

Document ingestion and preparation ends with the filtered term/document frequency matrix. This is an array whose entries correspond to the number of times each token appears in each separate document minus undesirable entries such as stop words, hapaxes (terms that appear only once) or other very high or very low-frequency tokens.

3.3.2 Latent Semantic Analysis

Latent Semantic Analysis (LSA) is a natural language processing technique used to analyze relationships between a set of documents and the terms contained in the documents by producing compressed sets of topics related to the documents and the terms.

Following the construction of the frequency matrix, the LSA method computes a low-rank approximation to the term-document matrix. The result of this rank-lowering is that several matrix dimensions are combined to depend on more than a single term, mitigating the problems of identifying synonymy[1], and developing topical clustering in the high dimensional space. Rank-lowering also partially mitigates the problem of polysemy[2] by weighting terms that commonly occur together in similar dimensions.

Rank-lowering can be accomplished in one of many ways, although the preferred method is to utilize a Singular Value Decomposition (SVD) of the frequency matrix. By selecting the k largest singular values from this decomposition, along with the singular vectors, we

[1]synonymy - different words have similar meaning or convey the same ideas, for instance, 'physician' and 'doctor'

[2]polysemy - same word conveys different meanings, for instance, the term 'bed' has a different meaning in each of these phrases: 'river bed' versus 'sleeping bed'

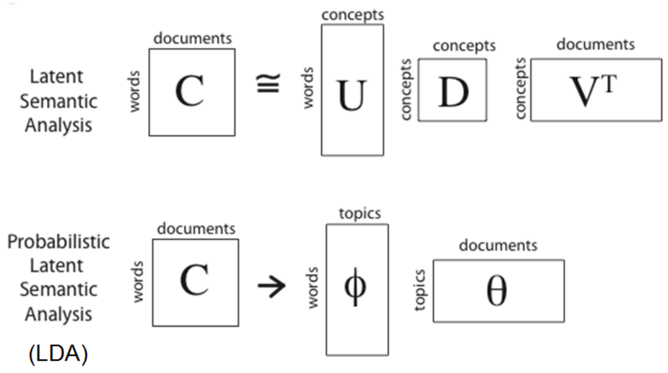


Fig. 3. Generalized equations for LSA and LDA solutions. (Image courtesy of David Robinson, Sandia National Laboratories.)

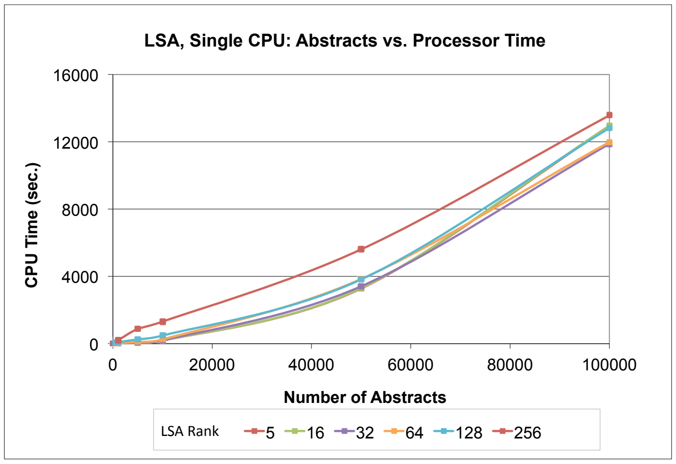


Fig. 4. Single processor runs of the LSA pipeline with increasing number of abstracts. Some non-linear behaviour is present in increasing numbers of abstracts, while rank designation has little affect on timing.

obtain the rank k approximation of the original frequency matrix with the smallest error. This approximation can be cast as a "clean" version of the term/document matrix with low-importance noise removed. Documents can be compared with one another by calculating distances in this low-dimensional space.

In ParaText, LSA is implemented in two phases. The first is a weighting phase in which the contents of the term/document matrix are transformed to represent term importance or Shannon information instead of raw occurrence counts. We do this with a simple parallel filter. The second is the singular value decomposition, computed using the distributed block Krylov-Schur method from the Anasazi package in the Trilinos solver library [22]. We compute document similarity scores using the cosine similarity measure between document vectors in the SVD results.

In addition to Latent Semantic Analysis, the ParaText application within Titan contains a sister executable for performing Latent Dirichlet Allocation (LDA) analyses. These two types of analyses are similar in their overall function but provide some differences in the resulting outputs.

3.3.3 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA), a probabilistic alternative to LSA, has gained currency in recent years as a topic modeling algorithm. LDA was first proposed by Blei et al. [3] as a topic model with greater expressive power than LSA. LDA models a document as a mixture of topics where each topic is a probability distribution over the vocabulary. Proponents of LDA argue that these topics are more easily interpreted than the singular vectors constructed by LSA. A high-level comparison of the LSA and LDA approaches can be found in Figure 3 and in Table 1.

We use LDA to model a corpus of documents as mixtures over topics by inverting this generative approach: instead of creating documents from a known set of topics, we use Bayesian inference to learn the topic parameters from a set of training data (documents). We have implemented a stochastic version of LDA based on collapsed Gibbs sampling [15] in Titan with both serial and distributed-memory parallel versions. We use this implementation in all phases of our project including scaling studies and extracting topic models from Pubmed.

3.4 Scaling Studies of Text Analysis Algorithms

A major goal of this project was to demonstrate the feasibility and scalability of the LSA and LDA pipelines within the Titan Scalable Informatics toolkit. We used the ParaText application to run scalability studies with both LSA and LDA using the PubMed corpus. In this section we discuss the results of these studies.

3.4.1 Comparisons between LSA and LDA

All scalability studies outlined in this section were completed on the Red Sky platform described in Section 3.1. As described previously,

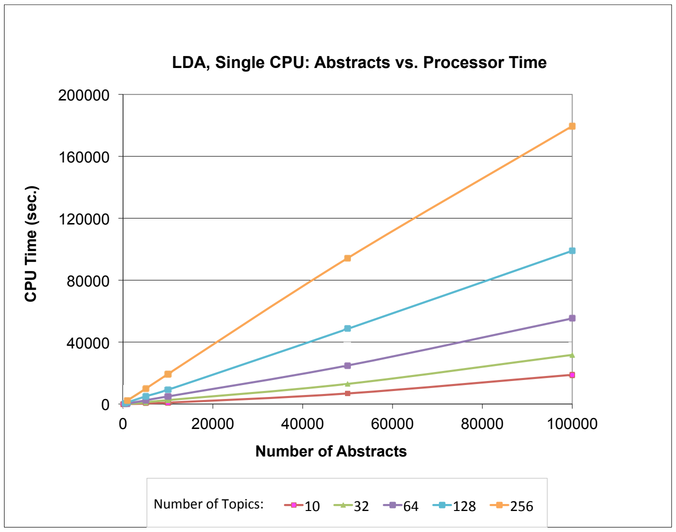


Fig. 5. Single processor runs of the LDA pipeline with increasing number of abstracts. Timing behaviour appears perfectly linear for increasing numbers of abstracts, while slope of the timing is highly dependent on number of topics chosen. On a single processor LSA has a large speed advantage due to its highly optimized linear-algebra core.

Table 1. Feature comparison of LSA and LDA.

	LSA	LDA
Mathematical Model	Vector Space	Probabilistic (Bayesian)
Input	Term x Doc matrix Number of Topics	Term x Doc matrix Number of Concepts
Output	Term x Topic matrix Topic Weights Topic x Doc matrix	Term x Concept matrix Concept x Doc matrix
Typical Implementation	Singular Value Decomposition	Markov Chain Monte Carlo or Variational Inference
Handling New Documents	Recompute from scratch (some work on fast updates)	Quick, incremental update
Deterministic?	Yes - SVD has one correct answer	Theoretically, no; in practice, yes
Polysemy	Limited support	Yes

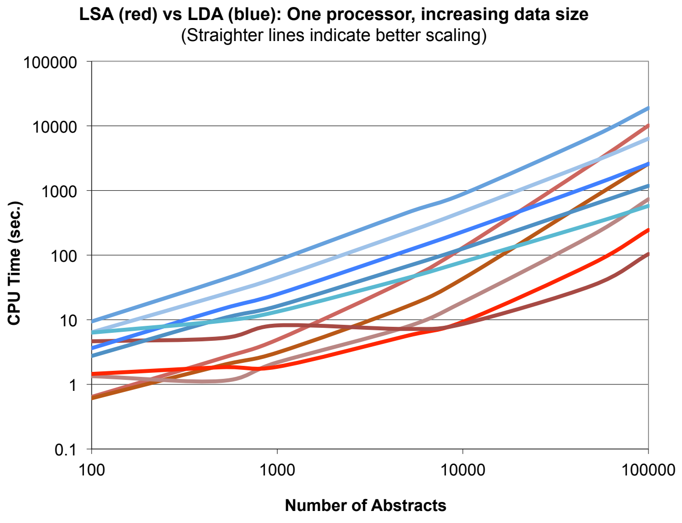


Fig. 6. Comparison of LSA (red) and LDA (blue) algorithms showing increasing work load vs. time to completion. Shown here are (in ascending order) timings for 1, 2, 4, 8 and 16 processors.

each Red Sky node consists of two quad-core processors sharing 12Gb of memory. Apart from minor OS overhead, each processor core thus has 1.5Gb of memory available to aid computation unless special runtime parameters are specified to allocate additional memory to a given core. In the remainder of

To begin the process of assessing scalability of the text analysis pipelines, we started with a relatively small workload on a single processor, gradually increasing work to assess performance. The results of these single-processor runs for LSA and LDA are shown in Figures 4 and 5. We observe that LSA demonstrates some non-linear scaling with increasing numbers of abstracts. LDA remains (apparently) perfectly linear, but the slope of the line is dependent on the number of topics designated.

We now discuss similar studies in a multiprocessor setting. While serial studies speak to the asymptotic complexity of LSA and LDA, parallel scalability studies highlight the overhead of communication and coordination as work is parceled out.

Work Scaling: In our first parallel test we measure execution time for a given number of processors as we increase the amount of data per processor. This is equivalent to the running test shown in Figures 4 and 5 several times, each time with a different CPU count. We show the results in Figure 6. The key insights from this test are that (1) our implementation of parallel LDA is slower than our parallel LSA code for small data sizes and (2) LSA appears to incur spurious communication costs for lower numbers of abstracts and higher processor counts. This is not unreasonable: such anomalies can easily be caused by transient OS or I/O overhead, especially with short overall execution times that are more visibly affected by small-scale noise.

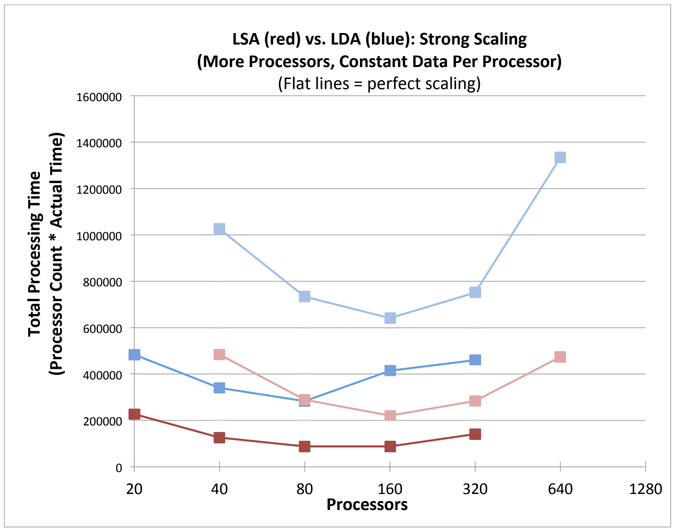


Fig. 7. Strong Scaling: The total work is held constant while the number of processors is increased. The graph indicates a sweet spot between reduction in computation time due to increased numbers of processors to do the work and the communication costs between the increasing number of processors.

Weak Scaling: In high-performance computing, "weak scaling" is the notion that if x processors can handle y items in z hours, $2x$ processors should ideally be able to handle $2y$ items in the same z hours. We tested this by choosing a data set (between 1,000 and 100,000 abstracts) and measuring execution time for both LDA and LSA as we varied the processor count from 1 to 1,000. The results are in Figure 8. Perfectly horizontal lines would represent ideal scaling. An upward curve indicates increasing overhead from inter-process communication. This, in turn, can be caused by congestion in the system interconnect as well as increasing amounts of data required for synchronization.

We see evidence of good scaling on larger amounts of data (10,000 abstracts and up). Communication overhead grows to dominate execution time for small data sizes with LSA more strongly affected than LDA.

Strong Scaling: In contrast with weak scaling, "strong scaling" is the notion that if a task can be completed in z hours with x processors, a system with $2x$ processors could complete the same task in (ideally) $\frac{z}{2}$ hours. That is, we choose a data size and hold it constant while increasing the processor count to solve the problem in less time. Ideally, a strong scalability study allows a user to specify a problem size and a desired time to solution and then look up the amount of hardware necessary to meet those constraints.

We show results in Figure 7. A horizontal line would indicate perfectly linear scaling. Instead we observe two phenomena. At low processor counts, each CPU core is handling enough data that it begins to bog down. We speculate that this comes from overload in the memory hierarchy. For high processor counts, execution time begins to rise again as each processor spends less of its time doing work and more of its time communicating its results. In the middle we find a sweet spot that indicates a balanced tradeoff between communication costs and the capacity of a single node.

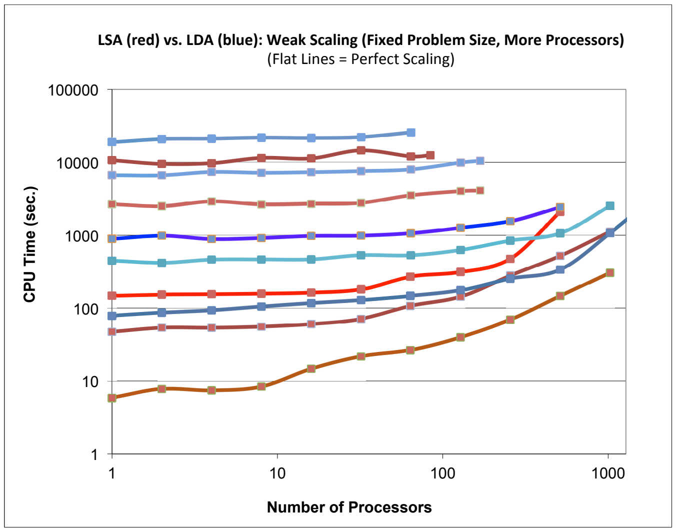


Fig. 8. Weak Scaling: Work per processor is held constant (in terms of the number of abstracts per processor) while the number of processors is increased. Perfect scaling would be shown by horizontal lines, while increasing slope is indicative of increasing communication costs between processors.

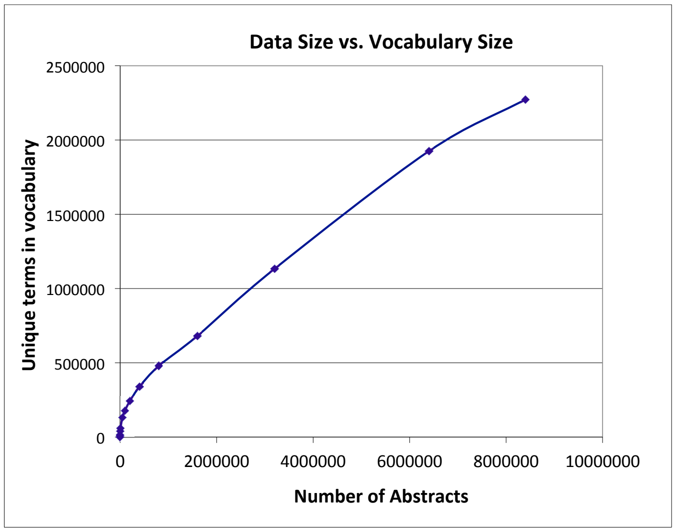


Fig. 9. Conventional wisdom indicates that the number of unique terms in a document collection begins to tail off around 200K terms. The PubMed collection shows the number of unique terms continues to climb with the number of abstracts beyond 2M terms.

Unexpected Verbosity: While collecting data for the scaling studies, we found that runs on larger data sets frequently ran out of memory and failed far before the limits we anticipated. We determined that this memory was being consumed by the term/topic array. We did not expect this to be a bottleneck. Conventional wisdom holds that the number of unique terms in a natural language corpus should approach a maximum, perhaps around 200K, and would thereafter remain fixed no matter how many documents were added. However, the PubMed collection shows the number of unique terms continues to climb more or less linearly with the number of documents as shown in Figure 9. This caused one of the large state arrays to grow linearly with the number of documents instead of remaining constant as we had hoped. In order to complete these larger runs, we had to allocate more memory to each processor and reduce the number of cores used per node to account for the larger term dictionaries that were being used by each processor. The longer-term solution to the problem was to filter lower-frequency terms out of the data set during preprocessing. We will discuss this further in Section 3.4.2.

3.4.2 Low memory studies of LDA

Memory consumption for both LDA and LSA proved to be a major limiting factor in our scaling studies. Since each node of Red Sky is outfitted with 12GB of main memory and 8 processor cores, we can use no more than 1.5GB per process if we want to take full advantage of the CPU. When we began our first full-scale LSA and LDA runs using all 17 million articles and a relatively low rank we found that we needed more than half of the total memory available on each node. This limited us to one CPU core per node out of the 8 available.

On investigation, we discovered that the bottleneck was the dense *term* $\times$ *topic* matrix (right singular vectors for LSA, ϕ for LDA) matrix. As shown in Figure 9, the number of unique terms in PubMed grows as a nearly linear function of the number of articles. The dictionary for all 17 million articles comprises 2.2 million distinct words, of which over 900,000 are $\mathbf{t}_h$apax legomena, i.e. only occur a single time in the entire corpus. These hapaxes compose 0.1% of the 1.2 billion words in the corpus. Because of the statistical nature of LDA, such infrequently-occurring terms will have no substantial impact on the results. Pruning the dictionary to eliminate terms that occur fewer than 20 times, ever, or in fewer than 5 documents reduces the size of the dictionary to 350,000 words but only removes 0.3% of the total words in the corpus. This improves the memory footprint dramatically. We also implemented a low-memory version of LDA to further reduce this resource footprint. In this variant, we do not store the θ and ϕ matrices explicitly but instead construct them directly from internal state data when writing the output to disk. This limits our LDA process to taking a single sample from the solution space. We feel that this is not a serious drawback: taking a single sample (as opposed to merging multiple samples) is common practice in the LDA literature.

Pruning the dictionary by removing low occurance terms reduced the LDA memory footprint by roughly 85%. Using the low-memory LDA variant gained another factor of 2 improvement. Without these modifications, the full-scale LDA runs used for the topical clustering would have taken three days of runtime using more than 25% of the nodes on Red Sky. Such a large job on an already heavily-used system would take one to two weeks to schedule (i.e., will sit idle in the queue waiting for enough processors to free up to begin running). After our modifications, our largest LDA process took three hours on 52 nodes (just over 2% of the machine) and was scheduled to run after less than two hours in the queue.

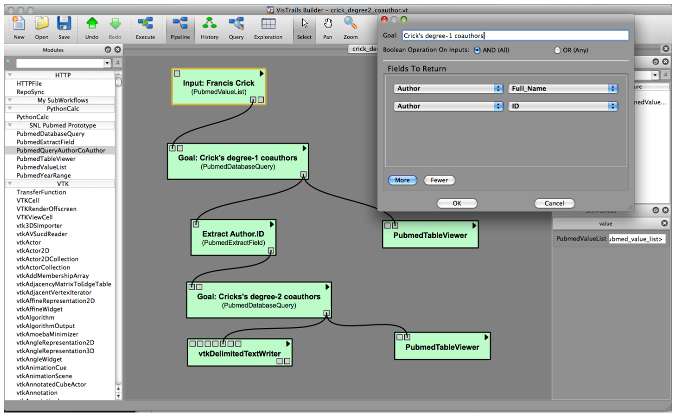
4 Focused Dataset Exploration

In addition to our investigation of the scaling properties of LSA and LDA, we produced several prototype explorations of the content of the PubMed data set. Each prototype was focused on a different question that arises during typical analysis and exploration. In this section we describe these prototypes in the context of their driving questions.

4.1 PubMed+VisTrails: Tracking the Workflow

In many analysis scenarios, the answer to a question of interest is incomplete without an accompanying pedigree. This pedigree must allow inspection of the chain of reasoning and evidence that support the analyst's conclusions all the way from source data to final report. In the past this has often been a manual process aided by an analyst's hand-written notes and recollections. We believe that this task should be automated as much as possible.

The VisTrails framework provides exactly this provenance through workflow tracking. We implemented a query-construction tool that enabled rapid interrogation of the database through questions that built on those already asked.



Goal: Crick's degree-1 coauthors

Boolean Operation On Inputs: [✓ AND (All)] OR (Any)

Fields To Return

Author	Full_Name
Author	ID

PubmedValueList ubmed_value_list>

Fig. 10. Vistrails application with a value list open for user input to start a search query.

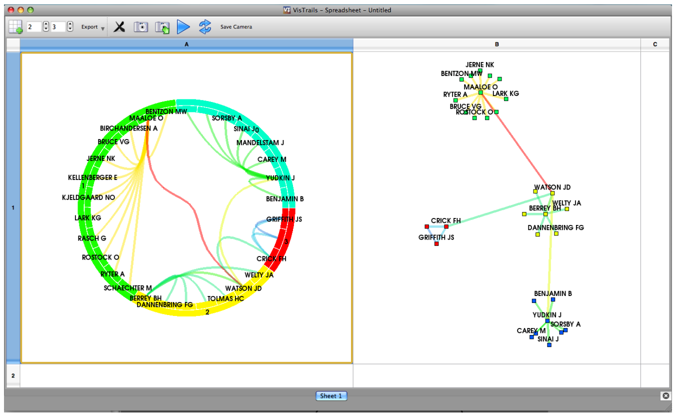
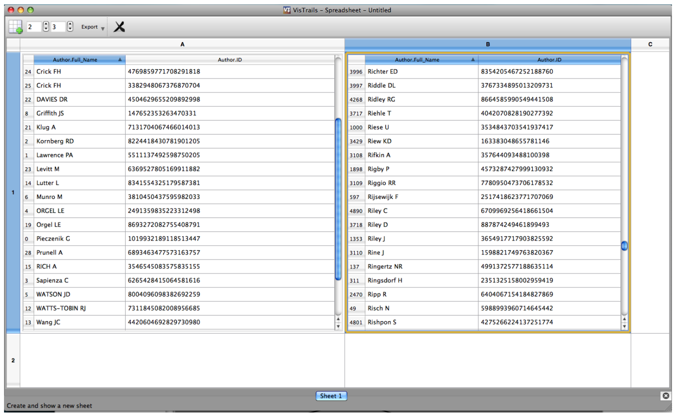


Fig. 12. Vistrails application tree ring and graph views for Francis Crick's degree one coauthors.


VisTrails – Spreadsheet – Untitled
2
3
Export
A
B
C
1
2
Sheet 1
Create and show a new sheet

	Author.Full_Name	Author.ID
24	Crick FH	4769859771708291818
25	Crick FH	3382948067376870704
22	DAVIES DR	4504629655209892998
8	Griffith JS	147652353263470331
21	Klug A	7131704067466014013
2	Kornberg RD	8224418430781901205
1	Lawrence PA	5511137492598750205
23	Levitt M	6369527805169911882
14	Lutter L	8341554325179587381
6	Munro M	3810450437595982033
4	ORGEL LE	2491359835223312498
19	Orgel LE	8693272082755408791
0	Pieczenik G	1019932189118513447
28	Prunell A	6893463477573163757
15	RICH A	3546545083575835155
3	Sapienza C	6265428415064581616
5	WATSON JD	8004096098382692259
12	WATTS-TOBIN RJ	7311845082008956685
13	Wang JC	4420604692829730980

	Author.Full_Name	Author.ID
3996	Richter ED	8354205467252188760
3997	Riddle DL	3767334895013209731
4268	Ridley RG	8664585990549441508
3717	Riehle T	4042070828190277392
1000	Riese U	3534843703541937417
3429	Riew KD	163383048655781146
3108	Rifkin A	357644093488100398
1898	Rigby P	4573287427999130932
3109	Riggio RR	7780950473706178532
597	Rijsewijk F	2517418623771707069
4890	Riley C	6709969256418661504
3718	Riley D	887874249461899493
1353	Riley J	3654917717903825592
3110	Rine J	1598821749763820367
137	Ringertz NR	4991372577188635114
311	Ringsdorf H	2351325158002959419
2470	Ripp R	6404067154184827869
49	Risch N	5988993960714645442
4801	Rishpon S	4275266224137251774

Fig. 11. Vistrails application spreadsheet output of a degree one coauthor query for Francis Crick.

4.1.1 Interface Overview

As discussed in Section 2.3, VisTrails provides a graphical interface for constructing data flow pipelines comprising sources, transformations and sinks. While this model is most obviously suited for data flow architectures such as VTK and Maya, it is equally applicable to database exploration with only a minor change in perspective. We need only three basic modules plus a small amount of syntactic glue.

- *Value List:* This module is the "source" for our data flow. It holds a list of values that form the keys for a query as well as the name of a field in which to search for those keys. Values can be arbitrary strings, numeric or alphabetic, to match any attribute in the database such as author names, journal titles, subject keywords or date ranges. At the beginning of an exploration the user enters values manually into one or more value lists. Figure 10 shows an example of a value list module.

- *Query:* This module is the "transformation" for our data flow. A query module takes one-to-many value lists as input. Its parameters are the type of combination for the input constraints (AND and OR) and the fields to return from the query.

- *Display/Output:* This module is our "sink". It takes the output from a Query module and displays it in a spreadsheet (see Figure 11) or tree ring view (see Figure 12). Alternately, the user can route output into an export module that writes images or structured text to disk.

The only piece still missing is the ability to recycle the output of one query as one or more of the inputs to another query. We achieve this by having the Query module output automatically-populate Value List modules. We also provide a simple "select field" module for the common case where a query returns multiple attributes for each database row.

These components are sufficient to express questions of the form "What values of [attributes] occur in articles for which [attribute1 is X, Y or Z] and/or [attribute2 is A, B or C] and/or [more constraints...]?". This highlights the central role of articles in the PubMed data: structural relationships such as co-authorship and publication in a single journal are exposed through chains of co-occurrence at the article level. This is sufficient for questions like the following:

- Who are the coauthors of Francis Crick to within 2 degrees of separation?

- In what languages (and with what frequencies) do those coauthors write?

- In what countries are the journals in which they publish?

- What are the journal titles and article titles for the publications in Spanish?

By linking the output of one query to the input of the next, an analyst can greatly increase his or her ability to rapidly formulate and execute complex, compound queries. The example above took about half an hour for a subject matter expert working with a software developer. The majority of that time was spent studying and interpreting the results: actual execution time totaled under 5 minutes. At every step the workflow was preserved as an artifact that could be annotated, replayed, saved, restored and used as a basis for future work.

4.1.2 Implementation Notes

We made extensive use of PyQt [21] and Titan (through automatically generated Python wrappers) when building the PubMed Database Explorer. When we began, most VisTrails modules were oriented toward fixed-function transformations with a small number of numeric or Boolean parameters. While VisTrails is able to automatically generate configuration tools for such modules, we found that we needed more targeted and customized interfaces for our use case. We gratefully acknowledge the support of the VisTrails development team in helping us find the correct approaches for this customization.

4.2 Topical Clustering

In addition to the LSA and LDA scaling studies, we used LDA to compute a topic model and document clustering over all 17 million articles in the PubMed corpus.

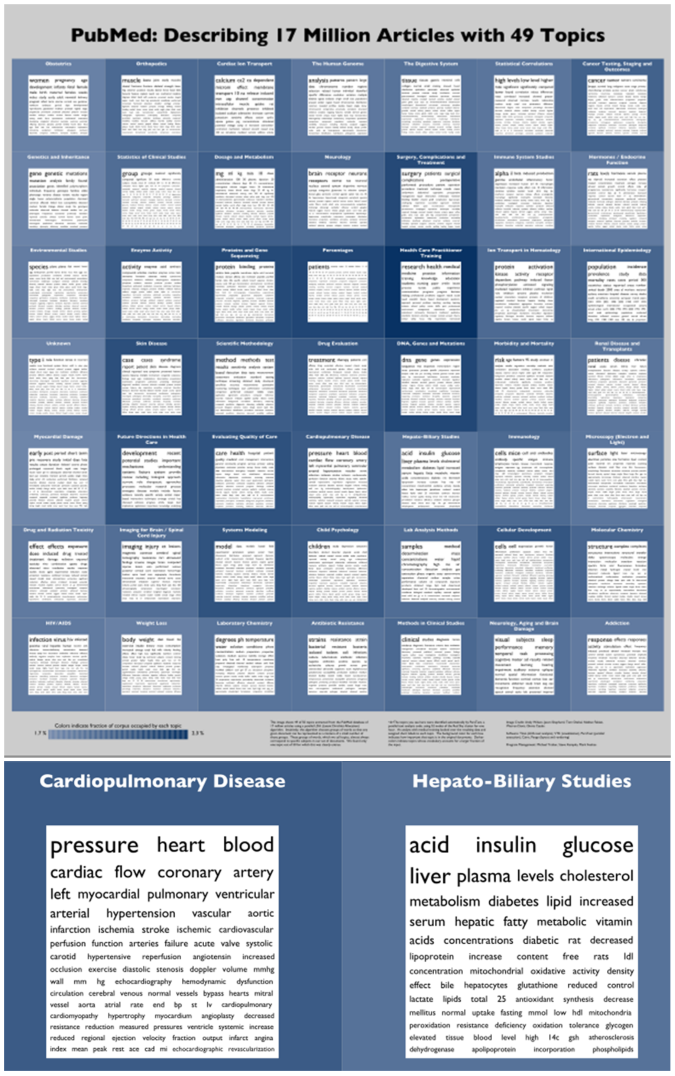

PubMed: Describing 17 Million Articles with 49 Topics
Obstetrics
Orthopedics
Cardiac Ion Transport
The Human Genome
The Digestive System
Statistical Correlations
Cancer Testing, Staging and Outcomes
Genetics and Inheritance
Statistics of Clinical Studies
Dosage and Metabolism
Neurology
Surgery, Complications and Treatment
Immune System Studies
Hormones / Endocrine Function
Environmental Studies
Enzyme Activity
Proteins and Gene Sequencing
Percentages
Health Care Practitioner Training
Ion Transport in Hematology
International Epidemiology
Unknown
Skin Disease
Scientific Methodology
Drug Evaluation
DNA, Genes and Mutations
Morbidity and Mortality
Renal Disease and Transplants
Myocardial Damage
Future Directions in Health Care
Evaluating Quality of Care
Cardiopulmonary Disease
Hepato-Biliary Studies
Immunology
Microscopy (Electron and Light)
Drug and Radiation Toxicity
Imaging for Brain / Spinal Cord Injury
Systems Modeling
Child Psychology
Lab Analysis Methods
Cellular Development
Molecular Chemistry
HIV/AIDS
Weight Loss
Laboratory Chemistry
Antibiotic Resistance
Methods in Clinical Studies
Neurology, Aging and Brain Damage
Addiction
Colors indicate fraction of corpus occupied by each topic
1.7 %
3.3 %
Cardiopulmonary Disease
pressure heart blood
cardiac flow coronary artery
left myocardial pulmonary ventricular
arterial hypertension vascular aortic
infarction ischemia stroke ischemic cardiovascular
perfusion function arteries failure acute valve systolic
carotid hypertensive reperfusion angiotensin increased
occlusion exercise diastolic stenosis doppler volume mmhg
wall mm hg echocardiography hemodynamic dysfunction
circulation cerebral venous normal vessels bypass hearts mitral
vessel aorta atrial rate end bp st lv cardiopulmonary
cardiomyopathy hypertrophy myocardium angioplasty decreased
resistance reduction measured pressures ventricle systemic increase
reduced regional ejection velocity fraction output infarct angina
index mean peak rest ace cad mi echocardiographic revascularization
Hepato-Biliary Studies
acid insulin glucose
liver plasma levels cholesterol
metabolism diabetes lipid increased
serum hepatic fatty metabolic vitamin
acids concentrations diabetic rat decreased
lipoprotein increase content free rats ldl
concentration mitochondrial oxidative activity density
effect bile hepatocytes glutathione reduced control
lactate lipids total 25 antioxidant synthesis decrease
mellitus normal uptake fasting mmol low hdl mitochondria
peroxidation resistance deficiency oxidation tolerance glycogen
elevated tissue blood level high 14c gsh atherosclerosis
dehydrogenase apolipoprotein incorporation phospholipids

Fig. 13. Progressive zooms of a topical clustering of the PubMed corpus. Topics were selected and hand-labeled by a subject matter expert into topical boxes. Each topical box contains a tag cloud of words sized by frequency of usage within that topic. (Continued in Figure 15.)

4.2.1 Topic Model

We used the parallel LDA implementation described in Section 3.4.2 to extract models with 50 and 100 topics. Each "document" comprised the title and abstract (if any) from a single article in the database. We used standard rule-of-thumb values for the α and β LDA hyperparameters (0.1 for β and $\frac{50}{K}$ for α where K is the number of topics) to encourage a topic model where documents concentrated their weight in relatively few topics.

We observe clear, comprehensible results at $K = 50$ as well as $K = 100$. Figure 13 shows examples of some of the topics returned. Examining the results has yielded one crucial insight: the mixtures of words resulting from LDA are not actually topics in the sense that humans expect: "a matter dealt with in a text, discourse or a conversation". Instead those pools are the *ingredients* for topics. For example, a paper discussing a new class of drugs to treat cancer will comprise frequent words from the LDA "topics" about statistics, cancer and proteins, among others. To obtain document groupings that humans might recognize as coherent we performed a further clustering step in the LDA topic space.

At this point, it is typical to run a cosine similarity using the results of the topic model to cluster similar documents. Unfortunately, this is an $O(n^2)$ algorithm. At full scale – that is, $n = 17$ million – that expense is intolerable. We, therefore, opted to reduce the space by running a multi-stage approach (described in the next two sections). This is accomplished by first reducing the space to $n = 1000$ by performing a preprocessing step that utilizes Gaussian expectation maximization[7]. This is followed by a hierarchical agglomerative clustering step. Using this approach we can reduce the runtime and memory costs to relative insignificance.

4.2.2 Preprocessing via Gaussian EM

Unfortunately, hierarchical agglomerative clustering is also an $O(n^2)$ algorithm, requiring all the pairwise cosine similarities to complete the agglomeration process. It is typical, however, for many of the initial agglomeration steps to merge documents that are virtually indistinguishable, at least from the perspective of LDA. As we are trying to identify conceptually different document clusters, we would like to skip these preliminary agglomerations.

We began by running Gaussian expectation maximization (EM)[7] on the documents in the k-dimensional LDA result space. In this particular case we used the results from the LDA run where K=100. Our parallel EM implementation was built from Titan components and used a MapReduce algorithm running over MPI. We constrained the Gaussians in this case to be axis-aligned for simplicity and efficiency.

This preprocessing allows us to predict the number of pairwise similarity calculations between nearly indistinguishable documents. We can skip these calculations, reducing the number of calculations required in the next stage from 2.89×10^{14} to 1.0×10^{6} – a reduction of eight orders of magnitude. Depending on the data, this efficiency gain will be somewhat reduced by the overhead of the preprocessing costs.

4.2.3 Hierarchical Agglomerative Clustering

Hierarchical agglomerative clustering seeks to merge observations, starting with the observations that are closest, as defined by some proximity measure. This process is straightforward when merging individual observations. However, as this agglomeration proceeds, it will eventually be necessary to agglomerate groups of these observations together. In this case, the proximity between these agglomerated clusters is not so clear. There are various ways of perceiving the proximity, or *linkages*, between clusters.

- *Single-linkage* defines the proximity between two clusters as the proximity of their closest points. This method will have a peculiar effect in the presence of outliers. Clusters can be deemed close to each other simply because they have outliers that are close, even if all the other members of the cluster are far apart. This process can be repeated many times, producing an agglomeration that might be counter-intuitive. This process is called *chaining*. For many applications, chaining is not a desirable effect.

- *Complete-linkage* defines inter-cluster proximity as the proximity of the farthest points. Like single-linkage, complete-linkage also suffers in the presence of outliers, although in an opposite manner. Clusters that would intuitively be consider close can be deemed far because of their outliers.

- *Average or centroid-linkage* defines the proximity of two clusters as the proximity of the centroids of the two clusters. This method is more robust than single and complete linkage in that the presence of outliers is mitigated by the other observations that compose the cluster.

- *Group-average-linkage* is the average proximity between the observations from one cluster to the observations in another. Similar to average-linkage, group-average-linkage is robust in the presence of outliers. In some ways, it is more 'democratic', in that every observation contributes directly to the final proximity, whereas with average-linkage, a 'representative' (centroid) is chosen. Generally, this may be unimportant, but could be a factor if the resultant cluster agglomerations are not hyper-ellipsoidal.

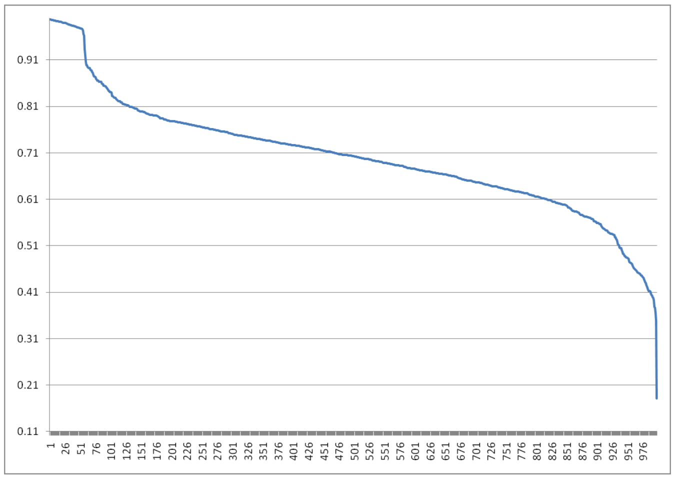


Fig. 14. Chart showing declining similarity scores between successive merges during the hierarchical agglomerative clustering phase. Values along the x-axis are number of merges, while values along the y-axis indicate the similarity between merged clusters.

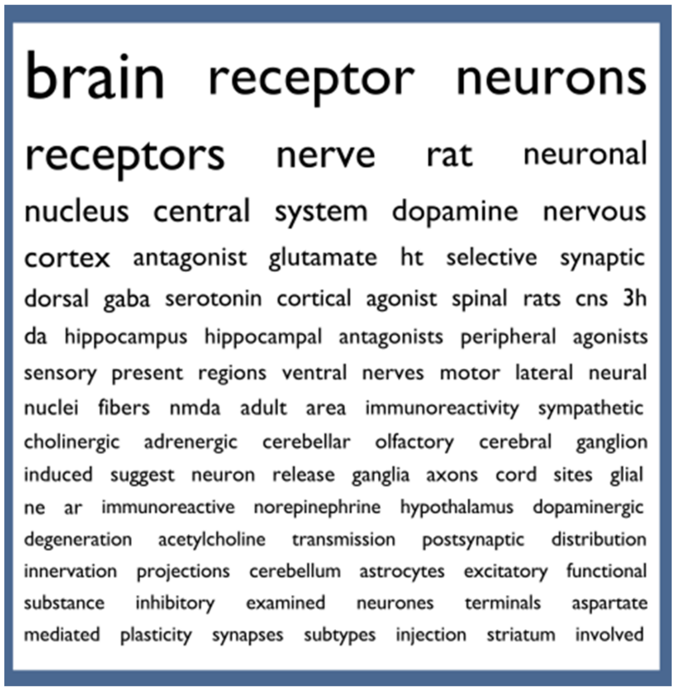

brain receptor neurons
receptors nerve rat neuronal
nucleus central system dopamine nervous
cortex antagonist glutamate ht selective synaptic
dorsal gaba serotonin cortical agonist spinal rats cns 3h
da hippocampus hippocampal antagonists peripheral agonists
sensory present regions ventral nerves motor lateral neural
nuclei fibers nmda adult area immunoreactivity sympathetic
cholinergic adrenergic cerebellar olfactory cerebral ganglion
induced suggest neuron release ganglia axons cord sites glial
ne ar immunoreactive norepinephrine hypothalamus dopaminergic
degeneration acetylcholine transmission postsynaptic distribution
innervation projections cerebellum astrocytes excitatory functional
substance inhibitory examined neurones terminals aspartate
mediated plasticity synapses subtypes injection striatum involved

Fig. 15. (Continued from Figure 13.) Progressive zooms of a topical clustering of the PubMed corpus. Topics were selected and hand-labeled by a subject matter expert into topical boxes. Each topical box contains a tag cloud of words sized by frequency of usage within that topic.

There are many other types of linkages available and the "best linkage" is often very application-dependent. We decided to use the group-average linkage, as it is more robust in the presence of outliers than single and complete linkage. In addition, its 'democratic' nature suggests that it may be more accurate in this domain than the 'representative' average-linkage algorithm. We have implemented a method of calculating group-average linkage that does not require recalculating the pairwise proximities between the elements of the merged clusters on each iteration, and therefore is as efficient as the three other linkages.

4.2.4 How Many Clusters?

An advantage of hierarchical agglomerative clustering is that it provides metrics as to the quality of our different clustering levels. Figure 14 shows the similarity value for each successive merge, starting with 1000 clusters of nearly indistinguishable documents and ending with one cluster containing the entire corpus. The smoothness of the curve between about 100 and 850 merges suggests that there may be more than one "right" number of clusters depending on the goals and knowledge of the user. This suggests that the clustering granularity should be exposed to the user.

4.2.5 Performance

We ran the LDA and Gaussian EM steps on the Red Sky capacity cluster. For 100 topics, LDA took 102 minutes using 340 CPU cores (43 nodes) using our low-memory variant. The Gaussian EM run took 413 iterations to converge over approximately 15 hours on 2048 cores (256 nodes). This reduces our efficiency gains from 7 to about 3 or 4 orders of magnitude.

One obvious possibility is to use simpler preprocessing techniques such as *k*-means instead of the highly expressive but computationally expensive Gaussian EM. This could allow us to salvage one or two orders of magnitude improvement, depending on the data. Nevertheless, we anticipate that clustering 17 million points in 100-dimensional space will always be an expensive operation.

The hierarchical agglomeration step has comparatively minimal hardware and software requirements. It took 3.1 seconds to run on a laptop using only one CPU core. The input to this process was the set of 1000 clusters that we computed using Gaussian EM. We have shown the final results for a single tag cloud cluster in Figure 15.

4.3 Coauthor Clustering

In contrast to the topical clustering, which derives from the *content* of the articles in the data set, we also chose to investigate the clustering of authors into communities. We derive these communities from the *structure* of the articles in the database.

We begin with a co-authorship graph. Each unique author in the database becomes a node in the graph. An undirected edge is present between nodes for any two authors who appear together on at least one paper. Edges are assigned weights equal to the number of co-occurrences of the corresponding authors.

We then use the weighted Clauset-Newman-Moore community-finding (wCNM) algorithm [2] to identify clusters of authors within this graph. Briefly, the wCNM algorithm proceeds as follows. First, each node V in the graph G=(V,E) is assigned to its own cluster. Second, we repeatedly merge the two clusters whose union produces the largest increase in the *weighted modularity* [2] of the induced graph. We continue merging communities until any further join operations would decrease the overall modularity. At that point we stop, label each connected component in the graph as a community, and output this labeling.

We chose to ignore nodes in the graph with degree higher than 50. These nodes almost always correspond to authors with very common names such as S. Kumar, R. Williams or H. Li. It is highly likely that these nodes represent several individuals with the same name who are incorrectly identified as being the same person. This is another of the consequences of "dirty" input data: PubMed does not provide a way to determine whether two different occurrences of any name represent the same individual. In the absence of a simple, robust way to perform this disambiguation, we choose to ignore high-degree nodes (individuals with more than 50 co-author links). We justify this ad-hoc decision as follows. First, highly-connected nodes have relatively low impact on the communities detected by the wCNM algorithm. Second, if a record is an agglomeration of publications by different people with identical names, keeping the node amounts to using known-to-be-incorrect data. Since this project did not encompass entity resolution or disambiguation, we felt that the cleanest approach was to ignore such nodes. In a situation where these methods were employed in ser-

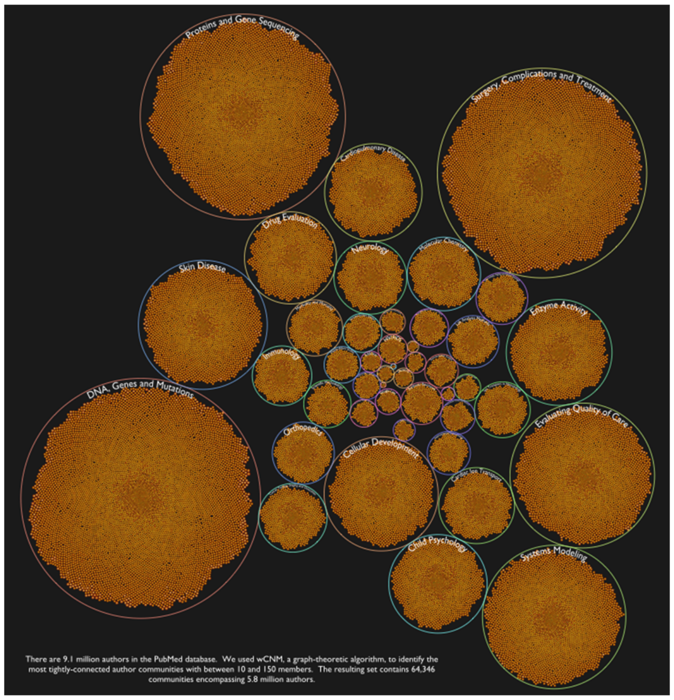


Fig. 16. Progressive zooms of a coauthor clustering of the PubMed corpus. Topics were selected and hand-labeled by a subject matter expert (large circles). Coauthors are clustered within each topic, and grouped by country. These cluster groups are then circle-packed within the larger topical clustering. (Continued in Figures 17 and 18.)

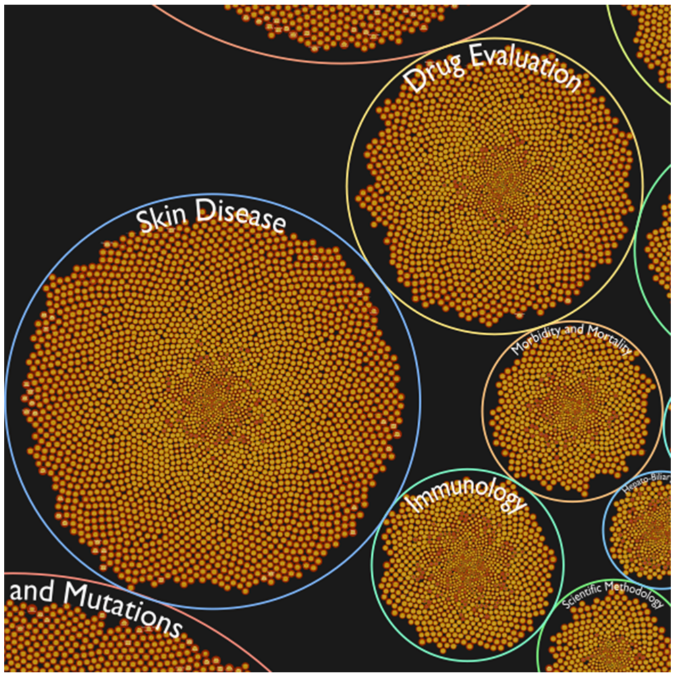


Fig. 17. (Continued from Figure 16.) Progressive zooms of a coauthor clustering of the PubMed corpus. Topics were selected and hand-labeled by a subject matter expert (large circles). Coauthors are clustered within each topic, and grouped by country. These cluster groups are then circle-packed within the larger topical clustering. (See also Figure 18.)

vice of an actual analysis this issue would require far more attention.

4.3.1 Results from wCNM

The co-author graph for PubMed contains 9.1 million vertices (one for each author in the database) and 84.3 million edges. We were able to run wCNM on one processor of a workstation with 64 gigabytes of main memory. The weighting stage took approximately two hours to complete. The hierarchical merge stage proceeded steadily for the first 7 million steps (out of 9 million total) and then slowed down dramatically for reasons we discuss later in this section. Since wCNM is a greedy algorithm that considers merges in decreasing order of importance, we believe these first seven million merges capture the most important community structures among the best-connected authors.

The wCNM algorithm produced many clusters with 1-10 members and a few clusters with hundreds of thousands of members. While this may or may not indicate a bug in the implementation, we know from sociology that coherent communities rarely have more than 150-250 members[12], and certainly do not grow to hundreds of thousands. In an attempt to derive socially meaningful clusterings, we stopped the wCNM merge process and declared a community whenever a join operation would increase a group's size above 150 members. We also imposed a lower bound of 10 members on the size of a research group for convenience.

Given these constraints, wCNM yielded 64,346 distinct clusters encompassing 5.8 million of the 9.1 million authors in PubMed. We further group these clusters by the most common category of vocabulary in their publications. Figures 16, 17 and 18 illustrate this clustering at increasing levels of detail.

The wCNM algorithm uses a precomputation phase to set edge weights. A priori edge weights can also be used to seed this process, but in this case, we computed derived weights for this clustering. These weights are based on two fundamental structures in the graph: 3-cycles (or“triangles”) and 4-cycles (or “rectangles”). Enumerating all triangles in the graph is a linear-time operation, and is implemented to run in parallel. In the PubMed experiments, however, we ran single-threaded on a 64GB workstation. The time spent in triangle-finding times is in the noise, compared to the other computations. Rectangle finding is much more challenging. Parallel algorithms exist for rectangle finding in MTGL, but can be memory intensive. A degree threshold is selected that eliminates consideration of vertices of sufficiently high degree. The current implementation also explicitly stores objects called “fake edges.” These are selected members of the set of edges in the complement of the graph.

By experimentation, we selected a degree threshold of 50, and computed wCNM weights on the PubMed graph allowing storage for fake edges of six times the size of the edge set of the graph. A direct relationship exists between these parameters: when we attempted to change the degree threshold to 100, the amount of space required for fake edges exceeded workstation memory. With feasible parameter settings (50, 6), our single threaded running time for rectangle enumeration was approximately two hours.

It should be noted that there is potential for running an order of magnitude, or more, faster on the Cray XMT. We did not attempt this during the PubMed experiment. However, we did attempt to implement another rectangle enumeration algorithm that promises to use much less space and can still be run with parallel processing. Unfortunately, during our experimentation we found that the new algorithm is designed to enumerate all rectangles without a degree threshold, and the number of rectangles is explosive without this limitation. Additional experimentation with degree thresholding is needed before this algorithm will be practical. This new algorithm, however, will enable efficient use of the XMT platform without excessive memory requirements. Edge weighting with this algorithm would be reduced from hours to minutes.

The second phase of the wCNM algorithm is a long sequence of agglomerative merging steps. All vertices initially reside in singleton communities. The CNM algorithm [6] selects successive merges in a greedy manner. Unfortunately, CNM is also known to produce un-

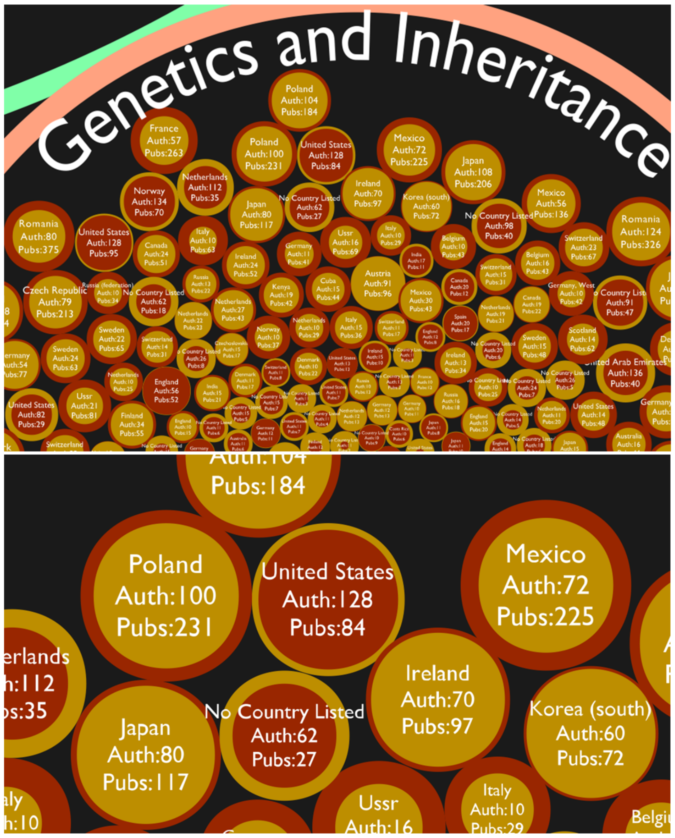
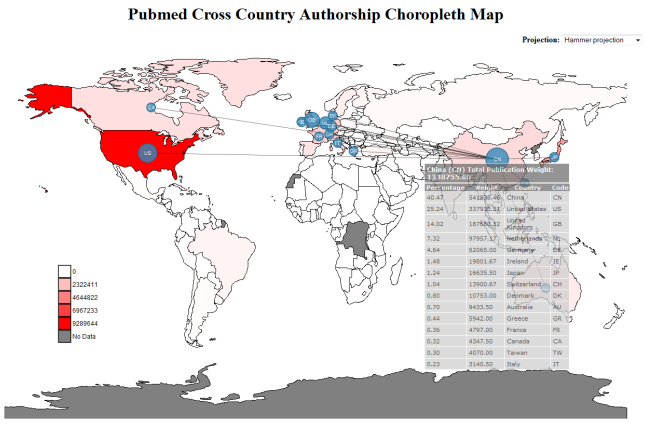


Fig. 18. (Continued from Figures 16 and 17.) Progressive zooms of a coauthor clustering of the PubMed corpus. Topics were selected and hand-labeled by a subject matter expert (large circles). Coauthors are clustered within each topic, and grouped by country. These cluster groups are then circle-packed within the larger topical clustering.


Pubmed Cross Country Authorship Choropleth Map

Projection: Hammer projection

China (CN) Total Publication Weight: 1338755.60

Percentage	Weight	Country	Code
40.47	541838.46	China	CN
25.24	337933.31	United States	US
14.02	187680.32	United Kingdom	GB
7.32	97957.17	Netherlands	NL
4.64	62065.00	Germany	DE
1.48	19801.67	Ireland	IE
1.24	16635.50	Japan	JP
1.04	13900.67	Switzerland	CH
0.80	10753.00	Denmark	DK
0.70	9433.50	Australia	AU
0.44	5942.00	Greece	GR
0.36	4797.00	France	FR
0.32	4347.50	Canada	CA
0.30	4070.00	Taiwan	TW
0.23	3140.50	Italy	IT

Fig. 19. Protovis webpage visualization of the Pubmed cross country authorship data. The country China is currently selected, which shows the directed graph of cross country authorship for this country.

balanced dendrograms, which can severely impact running time. To address this problem, Wakita and Tsurumi proposed a new factor in the objective function called a "consolidation ratio." This ratio penalizes unbalanced merges (e.g., those between a community with few vertices and one with many vertices). Our work in the PubMed experiment showed that while well-intentioned, Wakita and Tsurumi's consolidation ratio is unable to correct the unbalanced merging problem. Therefore, we augmented their idea by raising the consolidation ratio (a fraction between 0 and 1) to the fifth power, making the penalty for unbalanced merges extreme. Unfortunately, even this solution was inadequate for the PubMed dataset. Of nine million merges, the first (approximately) six million complete within two hours, but the remaining unbalanced merges dominate, rendering the completion of all merges a quadratic-time computation that is not feasible for efficient completion of the clustering.

Parallelizing the merging process, or developing an alternative method, remains open research. A potential and simple surrogate method is intriguing: simply sort the edges by wCNM weight and determine the sequence of merges based on the sorted order. Parallel sorting is optimized on the Cray XMT, thus lending some credance to this approach; however, this method remains to be tested and evaluated against the quality of the resulting solutions.

4.4 Cross Country Authorship Geo-Visualization

The articles in the Pubmed data set have an affiliation field where country of origin information can be extracted. The affiliation field is generally where the authors of the article list the institution they were associated with where the research was conducted. Each Pubmed article is associated with a journal where the article was published, and each journal in turn has country of origin information. By combining these two sources of country of origin information, we produced a directed graph where the graph nodes represent countries and the graph edges represent a weighted number of articles. The source of a directed graph edge is associated with the number of articles published by the source country in a journal affiliated with the target country of the graph edge.

To visualize the directed graph of Pubmed cross country authorship, we developed an interactive website using the Stanford Protovis [4] Javascript toolkit. A screen capture of the webpage is shown in Figure 19. The webpage is a geovisualization where countries are colored on a white to red scale of the weighted number of articles published by the country. Hovering the mouse over a country displays a table of the top collaborator countries associated with this country. Clicking the mouse on a country causes the directed graph to be drawn for the selected country and its collaborator countries. Countries not connected by this directed graph are made invisible to aid visual clarity. The size of the blue dots in the interior of each country represent the relative size of the weighted number of articles.

5 Discussion

The process of generating the visualization presented in the last section has been a learning process. In this section, we discuss several of the lessons learned from this project in terms of ongoing issues and items that should be addressed with each project.

5.1 HPC Resource Scheduling

We observed that HPC informatics jobs often have scheduling needs at odds with the way large parallel systems are managed for engineering and simulation workloads. Engineering projects often have timelines measured in months. Very large simulations have sufficiently large setup and planning costs and sufficiently long run times that they can be anticipated and scheduled far in advance. By contrast, information exploration applications often need a medium-to-large amount of processing power for a very short time and on very short notice.

The lesson we learned from this effort was that the hardware needs of our codes, especially memory, are of paramount importance. When we ran the scaling studies of LSA and LDA using larger data sets, we used all 12GB of memory on a single node for a single process. The other 7 CPU cores on that node sat idle. This bottleneck resulted in queue times of six to eight days for jobs that processed the entire PubMed data set. After we implemented low-memory LDA, the same jobs on the same data generally began execution within one hour.

5.2 Working With PubMed Data

Working with document collections that are not your own often presents some unique challenges. In this section, we discuss items pertinent to working with the PubMed document collection and database.

5.2.1 Database Organization

In the central PubMed repository operated by NIH, the *article* is the central entity around which everything else is organized. Each article has the following attributes:

- An article title
- An abstract
- A list of authors
- The journal issue in which the article appeared, including
  - The journal title
  - The journal's language
  - Country of publication
  - Volume and issue
  - Date of publication
- A list of subject descriptors (MeSH keywords)
- A list of investigators
- A list of funding sources (usually grants) that supported the work in the article
- A list of affiliations (but, see also Section 5.2.2)

People (authors and investigators) are identified by full names (forename, first, middle and last names, plus a suffix like Jr.). It is conventional in PubMed to identify an author with their initials plus family name. For example, the first two authors of this paper would appear in PubMed as AT Wilson and MW Trahan.

We use a star schema to represent PubMed data in our database. We assign globally unique identifiers to articles, authors, investigators, journal issues, subject keywords and funding sources. This raises the important issue of entity disambiguation. We will discuss this further in Section 5.2.2.

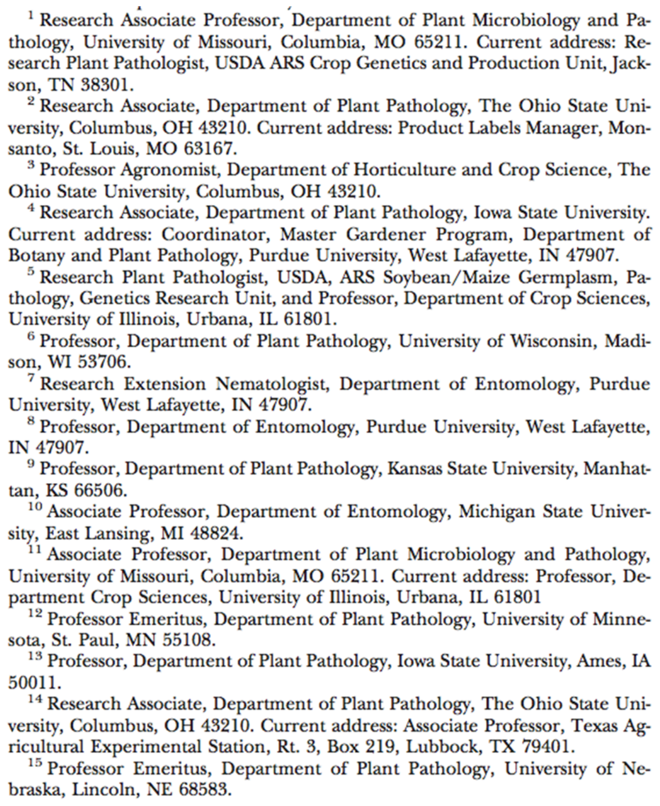
5.2.2 Cleaning the Data

Like many real-world data sets, the PubMed repository has properties that require human attention before unsupervised clustering and modeling algorithms may be applied. In this section we briefly discuss the issues in cleaning the data and rendering it amenable to efficient queries and analysis.

At a high level the PubMed data set is very well organized. We obtained the data in XML format along with a corresponding schema (http://www.nlm.nih.gov/bsd/licensee/data_elements_doc.html). This immediately removes much of the difficulty in data cleaning. We encounter problems, however, when we inspect the actual data values. Fields are often missing, including critical information like an article's title or authors. Many articles have title attributes such as "[no title supplied]" or "[title unavailable]". We chose to leave these unchanged in the database and filter them out at a later stage.

When fields in the schema are populated they can be difficult to read. We encountered a number of different character sets including ASCII, Latin-9 and at least three variants of Unicode (UTF-16 and UTF-8 in two different normalizations). We were able to deal with some of this, especially the differing Unicode representations, by processing the data using a Python script that had strong built-in support for Unicode. Even so, there remained records in the data set that required manual correction before they could be reliably parsed.

Last and not least, even when a field in the database is populated and is encoded in a well-supported character set, its contents may not



[1] Research Associate Professor, Department of Plant Microbiology and Pathology, University of Missouri, Columbia, MO 65211. Current address: Research Plant Pathologist, USDA ARS Crop Genetics and Production Unit, Jackson, TN 38301.
[2] Research Associate, Department of Plant Pathology, The Ohio State University, Columbus, OH 43210. Current address: Product Labels Manager, Monsanto, St. Louis, MO 63167.
[3] Professor Agronomist, Department of Horticulture and Crop Science, The Ohio State University, Columbus, OH 43210.
[4] Research Associate, Department of Plant Pathology, Iowa State University. Current address: Coordinator, Master Gardener Program, Department of Botany and Plant Pathology, Purdue University, West Lafayette, IN 47907.
[5] Research Plant Pathologist, USDA, ARS Soybean/Maize Germplasm, Pathology, Genetics Research Unit, and Professor, Department of Crop Sciences, University of Illinois, Urbana, IL 61801.
[6] Professor, Department of Plant Pathology, University of Wisconsin, Madison, WI 53706.
[7] Research Extension Nematologist, Department of Entomology, Purdue University, West Lafayette, IN 47907.
[8] Professor, Department of Entomology, Purdue University, West Lafayette, IN 47907.
[9] Professor, Department of Plant Pathology, Kansas State University, Manhattan, KS 66506.
[10] Associate Professor, Department of Entomology, Michigan State University, East Lansing, MI 48824.
[11] Associate Professor, Department of Plant Microbiology and Pathology, University of Missouri, Columbia, MO 65211. Current address: Professor, Department Crop Sciences, University of Illinois, Urbana, IL 61801
[12] Professor Emeritus, Department of Plant Pathology, University of Minnesota, St. Paul, MN 55108.
[13] Professor, Department of Plant Pathology, Iowa State University, Ames, IA 50011.
[14] Research Associate, Department of Plant Pathology, The Ohio State University, Columbus, OH 43210. Current address: Associate Professor, Texas Agricultural Experimental Station, Rt. 3, Box 219, Lubbock, TX 79401.
[15] Professor Emeritus, Department of Plant Pathology, University of Nebraska, Lincoln, NE 68583.

Fig. 20. Example of a common affilitations page with formatting. This is a screen capture of a PDF of one of the papers in the PubMed repository as originally published. Its structure is clear and evident and includes links back to the list of authors in the form of the superscripted numbers before each entry. Compare with Table 2.

be usable. The most prominent example of this in PubMed is the Affiliations attribute on each article that lists institutions for each author. In print-ready publications this information is typically organized into a list where each element is explicitly associated with one or more authors. See Figure 20 for an example. This structure is not present in the data retrieved from PubMed. We have instead a text dump of the affiliations list (see Table 2). In some cases a few simple heuristics suffice to separate these lists into their components. For example, addresses in the United States will often end with the name of a state or a ZIP code. Addresses elsewhere will often end with the name of a country. However, there are many more cases where this strategy fails. Since there is no strictly standardized format for affiliations, a single location may appear several times using different addresses (including permutations of a single address), different spellings and different languages. Even if we were able to overcome all these difficulties, one paramount problem remains: the mapping from institution to author is not present at all in the data.

In discussions with our colleagues we came to understand that PubMed is a remarkably clean example of a real-world data set. This leads us to conclude that data curation is a necessary fact of life rather than a marginal inconvenience. We assert that any research or analysis effort that will make heavy algorithmic use of real-world data must include data cleaning and ingestion explicitly in its plans by dedicating both time and money to the task.

5.2.3 Entity Resolution (Lack Thereof)

Worldwide, there are many different ways to write names and street addresses. Portions can be abbreviated or not ('J. F. Shepherd', 'J. Shepherd' or 'Jason Shepherd'), differently ordered (whether an individual's family name is written first or last), differently phrased

Research Associate Professor, Department of Plant Microbiology and Pathology, University of Missouri, Columbia, MO 65211. Current address: Research Plant Pathologist, USDA ARS Crop Genetics and Production Unit, Jackson, TN 38301 Research Associate, Department of Plant Pathology, The Ohio State University, Columbus, OH 43210. Current address: Product Labels Manager, Monsanto, St. Louis, MO 63167 Professor Agronomist, Department of Horticulture and Crop Science, The Ohio State University, Columbus, OH 43210 Research Associate, Department of Plant Pathology, Iowa State University. Current address: Coordinator, Master Gardener Program, Department of Botany and Plant Pathology, Purdue University, West Lafayette, IN 47907 Research Plant Pathologist, USDA, ARS Soybean/Maize Germplasm, Pathology, Genetics Research Unit, and Professor, Department of Crop Sciences, University of Illinois, Urbana, IL 61801 Professor, Department of Plant Pathology, University of Wisconsin, Madision, West Lafayette, WI 53706 Research Extension Nematologist, Department of Entomology, Purdue University, West Lafayette, IN 47907 Professor, Department of Entomology, Purdue University, West Lafayette, IN 47907 Professor, Department of Plant Pathology, Kansas State University, Manhattan, KS 66506 Associate Professor, Department of Entomology, Michigan State University, East Lansing, MI 48824 Associate Professor, Department of Plant Microbiology and Pathology, University of Missouri, Columbia, MO 65211. Current address: Professor, Department Crop Sciences, University of Illinois, Urbana, IL 61801 Professor Emeritus, Department of Plant Pathology, University of Minnesota, St. Paul, MN 55108 Professor, Department of Plant Pathology, Iowa State University, Ames, IA 50011 Research Associate, Department of Plant Pathology, The Ohio State University, Columbus, OH 43210. Current address: Associate Professor, Texas Agricultural Experimental Station, Rt. 3, Box 219, Lubbock, TX 79401 Professor Emeritus, Department of Plant Pathology, University of Nebraska, Lincoln, NE 68583.

Table 2. An example of affiliations as actually stored in the PubMed database. These entries are from the same paper as in Figure 20. Parsing this field into separate entries, parsing each entry, and associating entries with authors are all difficult problems in their own right.

('China' versus 'People's Republic of China') and written in different languages. This leads to the well-known challenge of resolving entities to determine whether two names or addresses refer to the same entity.

In this project we made only a minimal attempt to resolve entities. We corrected manually a few misspelled country names and identified authors using the semi-standard 'Shepherd JF' format. In the community-finding exercise, we mitigated the effects of many authors with the same name by ignoring any node (author) with more than 50 links (publications).

Ultimately, entity resolution requires knowledge of context. For example, graph algorithms based on citation networks can suggest that two names refer to the same individual because of similar sets of coauthors. This layer must also be exposed to the analyst, both for modification due to domain knowledge and awareness of the changes that have been made to the original data. Robust entity disambiguation and resolution remains a challenging research problem.

5.3 Large-Scale Visualizations

We have reached an era where the presentation of higher-order information is an essential step in exploring large data sets. A corpus of ten million documents contains too much information for a user to grasp without some sort of guide. Moreover, from a strictly pragmatic point of view, current high-end displays only contain 10-20 million pixels. Even if we could condense our representation to a singel pixel per document we would not have enough room to display everything at once.

We believe that the answer lies in offering tools for top-down as well as bottom-up exploration of the data in combination with a large display. A top-down perspective leads us toward topical decompositions, timelines, geographic summaries and community maps. A bottom-up structure suggests database searches organized around key words, specific names, and linkages through co-authorship or other affiliation. The use of a large high-resolution display allows us to take either of these approaches and show different views side-by-side so that a user can switch between them with eye movements that take only milliseconds instead of keyboard and mouse operations that take several seconds and demand conscious attention.

In our interactions with domain experts we have presented data on 30" LCD displays and 2-meter projection screens. The desktop monitor worked well when pairing one expert user with one domain expert. The projection screen was most valuable in allowing a group of analysts to discuss their thoughts and ideas without crowding around one desk. We continue to investigate how best to make use of large displays.

6 Future Work

In this section, we highlight several areas of future work and improvements. These efforts are currently on-going.

6.1 Larger Data Sets

In spite of the challenges posed by its scale, the PubMed data set is relatively small compared to many corpora of real-world interest. For example, most of the records in our data correspond to full papers containing several thousand words instead of a few hundred. Others such as the United States patent database can be even larger.

Such data sets pose three main challenges. First, they can be difficult to obtain. Because of the effort that goes into compiling very large databases, their owners often require substantial licensing fees for bulk access. Second, the basic infrastructure for working with document collections grows to require high-performance computing in its own right. Tasks such as computing a term/document frequency matrix become out-of-core operations best suited to a database. Third, interaction design becomes critical to usability as even the number of clusters grows too large for easy display.

6.2 Involving Human Factors in visualization design

Interest in understanding how visualizations are interpreted by users has been an area of growing interest for several years now with increasing ability to peer into the human brain. Additionally, understanding the workflows that are utilized and improving the efficiency in developing answers for specific users, along with enabling flexibility in these workflows is an area of increasing interest. These areas are known as human factors in the visualization and softward design processes.

Software is often developed in a prototype format to test algorithmic capability or in an attempt to allow a user access to functionality to determine how well it approaches problem solution or offers new insights. Unfortunately, prototypes are typically 'hardened' into products, and the user becomes the flexible operant in dealing with shortcomings in the product or design. Human factors studies attempt to correct this, using an iterative approach to design and coupling software designers with the user base, to bring more complete and usable solutions and longevity to products. However, this process can require extra investments of time and resources, and is often overlooked to the potential detriment of the user community.

The PubMed project described in this paper operated under a prototype development model. While many of the results hold promise of expanded efforts, future involvement with human factors specialists should be given increased priority.

6.3 Database alternatives

There has recently been substantial work in database-like system development for distributed clusters of commodity hardware (clouds). We would like to investigate the suitability of cloud-based databases built on top of frameworks such as Hadoop [26] and BigTable [5]. In these environments tasks such as frequency matrix construction and filtering are easy to express. In fact, token frequency counting is one of the standard examples of a MapReduce algorithm. However, the relational join becomes awkward, making tools such as the PubMed Database Explorer (described in Section 4) that relies on chains of join operations difficult to implement. It is not yet clear what the ideal backing store for all of these tasks would be. In addition to the algorithmic structure we must also consider price and power consumption.

6.4 Web interfaces

Fueled in part by cloud computing paradigms, accessing data and visualization from a single desktop machine is becoming a source of inefficiency for analytics-based tools. The model utilized by the Internet is fueling a need to have tiered architectures such that algorithms for large datasets can run on higher performance machines, while still fostering efficient access from commodity desktops and laptop computers. In this vain, web interfaces are quickly becoming the ubiquitous fronts to the user community for many of the advanced informatics-based visualizations. Unfortunately, this paradigm shift will come at some cost to historical development projects as the shift is made from historic GUI applications to applications that run within a common web browser. During the development of the functionality reported here, this shift was felt strongly. Converting all of the functionality to a standard web interface has some hurdles that still need to be addressed, and this is a source of ongoing future development.

7 Conclusions

The PubMed dataset is a reasonably large collection of documents detailing a research history of the medical community. Because of the size of the collection, finding tools for efficient search and summarization at varied levels of abstraction can be difficult and problematic. In this report, we have documented some initial informatic research to probe this document collection utilizing the Titan informatics toolkit. In some examples, we have coupled the Titan algorithms with the VisTrails application to provide improved user interaction, flexibility and provenance tracking capabilities to improve the experience for an end-user capability. We also presented results using high performance computing algorithms for data retrieval and analytics and provide some comparison and initial benchmarking for these capabilities to bound our expectations as the research continues forward. These high-performance methods have included ParaText, including LSA and LDA text analytics techniques, large graph clustering, and parallelized database querying. For each of these techniques, we have demonstrated a multi-resolution visualization of the results, where zoom-in capabilities demonstrate the scale of the data at each resolution. This research demonstrates the breadth of capabilities available within the Titan toolkit, and specifically demonstrates these capabilities at scales unmatched within the research community. Ongoing efforts for expanding and developing these capabilities continues.

Acknowledgments

The authors wish to thank Jon Berry for his expertise and time in development and execution of the topological clustering work on the coauthor graphs.

References

[1] The netezza database appliance architecture: A platform for high performance data warehousing and analytics. White Paper, 2008.

[2] J. W. Berry, B. Hendrickson, R. A. Laviolette, and C. A. Phillips. Tolerating the community detection resolution limit with edge weighting. *arXiv:0903.1072v2[physics.soc-ph]*, Mar. 2009.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[4] M. Bostock and J. Heer. Protovis: A graphical toolkit for visualization. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, pages 1121–1128, Nov/Dec 2009.

[5] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Trans. Comput. Syst.*, 26:4:1–4:26, June 2008.

[6] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70(6):066111–+, Dec. 2004.

[7] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Soc., Ser. B*, 39(1):1–38, 1977.

[8] I. S. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In V. K. R. Grossman, C. Kamath and R. Namburu, editors, *Data Mining for Scientific and Engineering Applications*, pages 357–381. Kluwer Academic Publishers, 2001. Invited book chapter.

[9] I. S. Dhillon and Y. Guan. Clustering large and sparse co-occurrence data. In *Proceedings of the Workshop on Clustering High-Dimensional Data and its Applications at the Third SIAM International Conference on Data Mining*, 2003.

[10] I. S. Dhillon, Y. Guan, and J. Kogan. Iterative clustering of high dimensional text data augmented by local search. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, 2002.

[11] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, Jan 2001.

[12] R. Dunbar. *Grooming, Gossip and the Evolution of Language*. Harvard University Press, 1998.

[13] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL2005)*, pages 363–370, 2005.

[14] A. Gilat. *MATLAB: An Introduction with Applications, 2nd Ed.* John Wiley and Sons, 2004.

[15] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl. 1):5228–5235, April 6, 2004.

[16] Institute for Statistics and Mathematics of the WU Wien. The R Project for Statistical Computing. `http://www.r-project.org/` [1 December 2010].

[17] Kitware, Inc. *The ParaView Guide, 3rd ed.* Kitware, Inc., 2008.

[18] Kitware, Inc. *The VTK User's Guide, 11th ed.* Kitware, Inc., 2010.

[19] Lawrence Livermore National Laboratory. VisIt: Visualize It in Parallel Visualization Application. `https://wci.llnl.gov/codes/visit` [29 March 2008].

[20] MathWorks. MATLAB Overview. `http://www.mathworks.com/products/matlab/` [1 December 2010].

[21] Riverbank Computing Limited. What is PyQt? `http://www.riverbankcomputing.co.uk/software/pyqt/intro` [1 December 2010].

[22] Sandia National Laboratories. The Trilinos Project. `http://trilinos.sandia.gov` [1 December 2010].

[23] J. G. Siek, L. Q. Lee, and A. Lumsdaine. *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley Professional, 2002.

[24] The Stanford Natural Language Processing Group. Named Entity Recognition (NER) and Information Extraction (IE). `http://nlp.stanford.edu/ner/index.shtml` [1 December 2010].

[25] VisTrails, Inc. Vistrails. `http://www.vistrails.org` [1 December 2010].

[26] T. White. *Hadoop: The Definitive Guide*.